Definire un template di classe `dList<T>` i cui oggetti rappresentano una struttura dati **lista doppiamente concatenata** (doubly linked list) per elementi di uno stesso tipo `T`. Il template `dList<T>` deve soddisfare i seguenti vincoli:

1. Gestione della memoria senza condivisione.

2. `dList<T>` rende disponibile un costruttore `dList(int k, const T& t)` che costruisce una lista contenente `k` nodi ed ognuno di questi nodi memorizza una copia di `t`.

3. `dList<T>` permette l'inserimento in testa ed in coda ad una lista in tempo $O(1)$ (cioè costante):

   - Deve essere disponibile un metodo `void insertFront(const T&)` con il seguente comportamento: `dl.insertFront(t)` inserisce l'elemento `t` in testa a `dl` in tempo $O(1)$.
   - Deve essere disponibile un metodo `void insertBack(const T&)` con il seguente comportamento: `dl.insertBack(t)` inserisce l'elemento `t` in coda a `dl` in tempo $O(1)$.

4. `dList<T>` rende disponibile un opportuno overloading di `operator<` che implementa l'ordinamento lessicografico (ad esempio, si ricorda che per l'ordinamento lessicografico tra stringhe abbiamo che `"campana" < "cavolo"` e che `"eccellente" < "ottimo"`).

5. `dList<T>` rende disponibile un tipo iteratore costante `dList<T>::const_iterator` i cui oggetti permettono di iterare sugli elementi di una lista.